9 Control structure methods

Description

As mentioned in section , the term *control structure* is used for statements that control the order of execution of other statements. These include conditional statements, loop statements and break statements

Our programming language has some predefined control structures like `leave`, `restart`, `if:then`, `if:then:else`, `for:to:repeat` and `cycle` as used in the previous chapters. `Leave` and `restart` are examples of break statements; `if:then` and `if:then:else` are examples of conditional statements; and `cycle` and `for:to:repeat` are examples of loop statements.

The control structures `leave`, `restart`, and `if:then` are primitives (built-in) of qBeta. The other control structures are defined by means of methods. A method defining a control structure is called a *control structure method* or just *control method* for short. A control structure method represents an abstraction over all the specific activities like a method and class and is thus also referred to as a *control abstraction*.

As a first simple example of a control method, we will show how to define the `cycle`-statement as used in previous sections like in the example below:

```
cycle
   if (today.isFirstDayOfQuarter) :then
      JohnSmithsAccount.addInterest
      ...
```

Here `cycle` is a method used as a super method of a *singular method object* – singular method objects are described below.

We define `cycle` in the following way:

```
cycle:
   inner(cycle)
   restart(cycle)
```

Execution of `cycle` implies that `inner(cycle)` is executed repeatedly. Execution of `inner(cycle)` implies that the statements in a possible submethod of `cycle` are executed.

For the above example this implies that the statement:

```
   if (today.isFirstDayOfQuarter) :then
      JohnSmithsAccount.addInterest
      ...
```

is executed forever unless a `leave`– or `restart` is executed within `...`.

Singular method objects

A statement of the form:

```
cycle
   if (today.isFirstDayOfQuarter) :then
      JohnSmithsAccount.addInterest
      ...
```

is actually an object descriptor *derived* from cycle, describing an invocation of a singular method object.

We have seen example of singular objects described using **`obj`**:

```
account_1010: obj
   owner: val "John Smith"
   balance: var float
```

As said in section , a singular object is a description of one specific object. It is not an instance of a class.

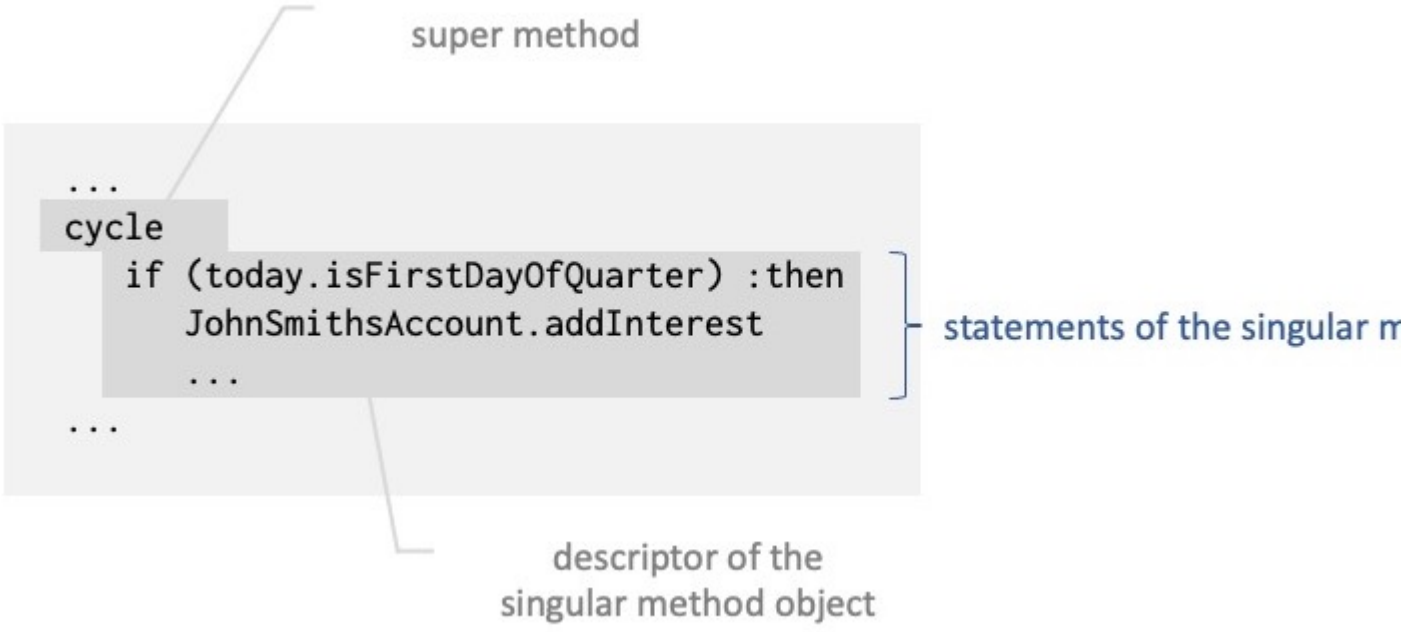
We may actually declare a similar singular *object* using `cycle` as follows

```
doIt: obj cycle
   if (today.isFirstDayOfQuarter) :then
      JohnSmithsAccount.addInterest
      ...
```

However, in most cases we will just use `cycle` as a statement and the statement:

```
cycle
   if (today.isFirstDayOfQuarter) :then
      JohnSmithsAccount.addInterest
      ...
```

is as mentioned a description of an invocation of a singular method object like `doIt` declared using **`obj`**. The differences are that as a statement the singular method object has no name; the `doIt` object is generated and executed as part of the enclosing object whereas the singular method object is generated and executed as a statement.



CycleSingularMethodDescriptor